HeinzelCluster: accelerated reconstruction for FORE and OSEM3D

St. Vollmar[1], C. Michel[2], J.T. Treffert[3], D. Newport[3], C. Knöss[1], K. Wienhard[1] and W.-D. Heiss[1]
[1]Max-Planck-Institut für neurologische Forschung, Gleueler Str. 50, 50931 Köln, Email: vollmar@pet.mpin-koeln.mpg.de
[2]PET Laboratory, Catholic University of Louvain, Louvain-la-Neuve, Belgium
[3]CTI Inc., Knoxville, TN, USA

I. Introduction

Iterative reconstruction techniques for reconstruction of Positron Emission Tomography (PET) data are usually too time consuming on most single processor machines that are affordable. This is especially true for the HRRT (High Resolution Research Tomograph) which demands sinogram dimension of unsurpassed size (presently one 3D data set consists of 2209 sinograms with 256 radial elements and 288 views), [1].

One strategy to drastically improve reconstruction time is the use of Fourier Rebinning (FORE), [4]: the 3D scan is transformed into the format of a 2D scan with 207 sinograms (in case of the HRRT) preserving the information of the 3D data. Thus the reconstruction problem is reduced to reconstructing independent 2D slices and offers a very convenient approach to cluster computing.

In a previous work we used this approach to scale down the reconstruction time with implementations utilizing RPC or Syngo[1] communication facilities on a Windows NT network of commodity PCs and with the first version of our dedicated reconstruction cluster (seven four-processor-systems, Intel PIII @ 700 MHz, 1 GB RAM, switched fast ethernet), [6].

These previous results have encouraged us to upgrade our dedicated cluster to Myrinet networking equipment [8] as we could identify fast ethernet bandwidth as the limiting factor (less so for special purpose network topologies with multiple fast ethernet cards per node). We think that we now have a good basis to tackle a more complex reconstruction method for cluster adaption: OSEM3D in the implementation of C. Michel [9]. This reconstruction method has produced the best results for the HRRT data so far and is more suitable for an adequate treatment of the sinogram gaps that result from the detector geometry of the HRRT.

We are also in the process of developing a complementary suite of tools to integrate cluster reconstruction for HRRT and ECAT7 data into our clinical routine.

II. Material and Methods

A. *Previous work based on RPC and Syngo*

The *BeeHive* package was mainly developed to utilize idle Windows NT user work stations for distributed computing of FORE-preprocessed sinograms, [5]. It consists of three components: (a) the "busy bees" (slaves) which are installed (automatically) on all NT machines, (b) the "BeeKeeper" component that is responsible for adding new bees to the beehive, and (c) the "QueenBee" (master) which distributes work among the "bees" and collects results. The idea is roughly this: have a single scheduling thread on the master maintain a list of jobs and a list of workers (a single thread to keep the algorithm simple and to avoid network bottlenecks/deadlocks). Distribute the jobs in a round-robin way, use redundancy where possible and allow for runtime reconfigurations of the slaves, have one single binary (90 KB) that is easy to maintain (network update, single click de/installation). As the original reconstruction problem by virtue of FORE-preprocessing is rendered "embarrassingly parallel", we did not see the need to rely on the usual message passing libraries (MPI, PVM, s. section *B*).

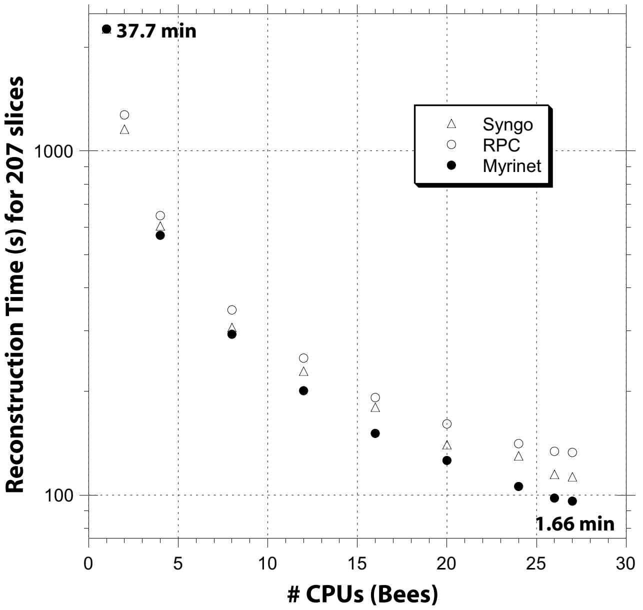

Figure 1: Influence of cluster size on reconstruction time for 207 slices (approx. 70 ML-EM iterations each), lin-log-plot, on our dedicated reconstruction cluster. Reconstruction time drops from 38 min in this example to well under 2 min for a full (FORE preprocessed) HRRT data set.

As described above, by using FORE the 3D-data sets are transformed ("rebinned") to independent 2D-data sets that can be reconstructed by any fast 2D reconstruction method, e.g. OSEM. Currently, we are using Cologne-HOSP, [3], that is based on Schmidlin's HOSP algorithm, [2], and has been port-

1 Syngo by Siemens AG is "a common language for software applications in the field of medical technology" (www.syngo.com).

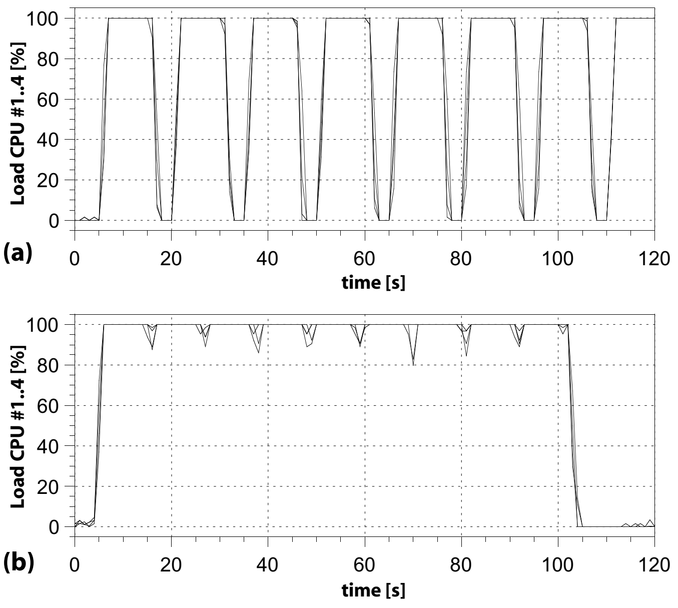

Figure 2: (a) CPU usage of one SGI 1450 node (four processors) of the *HeinzelCluster* during Syngo-based reconstruction of FORE-preprocessed data. This reconstruction involves 26 CPUs on seven nodes. (b) Same reconstruction job with Myrinet-GM-based implementation of *BeeHive*: observe the difference in idle time between productive intervals.

ed to *Solaris Sparc*, *IRIX*, *Linux x86*, *PowerMac* and *Windows NT*.

One disadvantage of the original RPC version is the polling approach of the scheduler which can create a lot of network overhead: each *Bee* is queried periodically for status information which can be a serious handicap when the cluster is inhomogeneous (too much time spent on slow machines). Syngo offers a powerful set of tools based on the ACE framework, [10], for implementing a pushing mechanism instead: the scheduler is notified about the status of a reconstruction bee automatically (proxy-return objects). Using Syngo also facilitates integration into the HRRT software environment (database access, reconstruction queues, visualization tools) as this is also based on the Syngo framework. Machines that only run a Syngo hosted reconstruction engine (backend) do not need a full Syngo distribution, a very small and easily maintainable subset suffices that does not create any overhead if not running.

We found that the simple RPC-based approach works fine for small homogeneous clusters (8 machines, speedup approx. a factor of 7) and that the more refined Syngo approach still gives good performance for homogeneous clusters of more than twice that size, s. Figure 1 and Table 1.

B. HeinzelCluster, BeeHive over Myrinet GM

The *HeinzelCluster* consists of seven SGI 1450 nodes, [15]: each has four Intel PIII Xeon processors @ 700 MHz, 1 MB L2-Cache, 1 GB RAM; all networked with full duplex fast ethernet on a Bay Stack 28115 switch; Myrinet M2M-PCI 64 A-2 cards and a Myrinet M2M SW16 switch and currently running Window NT. We decided against using Gigabit Ethernet as an alternative to Myrinet because this technology is limited to about 35 MB/s sustained data transfer by the IP stack handling, [11]. Myrinet is not much more expensive and gives much better performance: with our own Myrinet GM-based software (s.b.) we can transfer blocks of 52 MB size with a sustained rate of 130 MB/s and still expect more with the new generation "Myrinet 2000" hardware.

However, superior performance requires using special implementations of the message passing standard MPIch, [8], or adapting existing applications to GM, [12], a Myrinet proprietary message passing driver that is available for all major platforms.

We favor GM over standard message passing interfaces such as MPI and PVM (parallel virtual machine) because (apart from one scatter-gather function) we think that GM offers all the MPI functionality we need (s. *C*)—and on a standard NT system it just requires installing a single driver (like another regular ethernet card). As is pointed out in [13], MPI and PVM implementations handle network problems and hardware malfunctions usually quite ungracefully: GM is more robust and can resend lost packets automatically.

# CPUs (Bees)	Speedup Syngo	Speedup Myrinet
8	7.36	7.72
12	9.86	11.25
16	12.55	14.98
20	16.16	17.81
24	17.40	21.34
26	19.67	23.32

Table 1: Comparison of speedup *BeeHive* Syngo vs *BeeHive* Myrinet.

We have developed a little API that just requires linking one library and using one initialization sequence in order to send GM messages to any node of the cluster. The result is a Myrinet-GM based version of *BeeHive* that is fast, slim and conceptually very simple: on the *QueenBee* (master) and on each *Bee* (slave), i.e. on each of the seven cluster nodes, one GM event loop is running in its own thread and continuously polling for messages from other nodes (latency is known to be extremely good). Surprisingly, this creates very little overhead and allows for straight-forward event handling by implementing hooks that start appropriate worker threads, e.g. we send a message string that starts with a predefined identifyer (arbitrary integer): "4711"—receive sinogram of known dimensions (rest of message) and start reconstruction thread.

As can be seen from Fig. 2, with Myrinet-based *BeeHive* we visibly reduce idle time between "number crunching" periods.

C. OSEM3D acceleration

Porting the original Alpha version of OSEM3D (mostly ANSI C) to Windows NT was a straight-forward task. Our strategy to reduce reconstruction time (1 iteration for current HRRT data sets using the direct port on one Intel Pentium III@700 MHz takes about 11 h) consists of three steps: (a) parallelization for one node with four processors (shared memory), (b) higher level parallelization for the seven nodes to involve a reasonable number of CPUs (probably all) in the OSEM3D reconstruction, (c) platform specific optimizations.

We chose to work on (a) first, because a profile run clearly shows that more than 80% of the total wall time is spent inside the routines for forward and backprojection (which is the case for many iterative algorithms).

Thus we adapted the forward and back projection routines for multi-threaded execution with a wrapper routine (outer loop) leaving most of the original (unoptimized) engine untouched. This was an easy choice (and something Fortran compilers do by default on high perfomance platforms, [13]), e.g. the back projection algorithm is voxel driven so every thread takes over the voxels of 207/threads planes.

execution time [min]	PC PIII 600 Mhz	SGI 1450 1 Thread	SGI 1450 4 Threads	Compaq Alpha EV67
Normfac	259	212	73	76
1 Iteration	569	453	156	132
Total	828	665	229	208

Table 2: Multi-threaded implementation vs direct port. "Normfac" refers to a file of precomputed factors, s.b.

In case of the SGI 1450 running 4 threads, approx. 75% of the 229 min total time have been spent in either the forward or back projection routine, i.e. have been executed in parallel (which is acceptable for modifying only two routines and leaving almost everything else "as is"). The speedup of about 3 correlates nicely to the number of processors (4) and the time spent in parallel mode.

Apart from details in the thread synchronization (POSIX vs NT/2000), this approach should work for all multi-processor systems.

Approach (b) tackles parallelization with a much coarser granularity and is similar for the calculation of the "Normfac" data (precomputed factors needed only once for all iterations) and the main iteration loop. We propose to run the innermost loop (over the segments) of the pseudocode listing in Figure 3 in parallel. It follows that the theoretical speedup can not exceed the number of segments, which is 15 for the current configuration of the HRRT. After completion of the `isegment`-loop, the `cimage` volumes of all nodes must be collected and added (52 MB per cluster node) and for parallel execution of the next segment all nodes have to be synchronized by updating their local image buffer (another 52 MB per cluster node).

Finally, we believe that with (c) there is a significant potential for platform specific tuning we should investigate. However, we want to perform these optimizations last in order to stay as close to the original code as long as possible. Here the most relevant optimizations will be directly linked to the challenge of using the Pentium III's advanced (and often neglected) features more efficiently. First tests with the new Intel 5.0 compiler plugin for the Microsoft Visual 6 C++ compiler have been encouraging. In specific, we have successfully vectorized portions of code using compiler intrinsics (rather than Assembler code). With the Streaming SIMD extensions (SSE) of the PIII

```
for ( iter ) {
  for ( isubset ) {
    zero ( cimage )
    for ( isegment ) {
        forward3D ( image, &estimate )
        corrections ( trues, norm, atten,
                        &estimate )
        backproj3D ( estimate, &cimage )
    }
    image = image * cimage
  }
}
```

Figure 3: Pseudocode for OSEM3D (non-parallel) main loop. `image` and `cimage` are image volumes, `trues`, `norm` and `atten` hold the appropriate projection data for segment `isegment`.

that can give a speedup of up to 4 on single precision float operations. Furthermore, the PIII processor is rather dependent on proper alignment of operands for many of its more efficient operations (the Alpha processor is much more lenient, admittedly). All these changes will also be beneficial to potential Linux applications (on the Intel platform) of the code.

D. BeeQ, Vinci and clinical routine

In previous versions of *BeeHive* we used the *BeeKeeper* component for automatic network enabled update of the *Bee* service. With *BeeQ* we have extended the original concept: *BeeQ* is running as a service (similar to a Unix daemon) on every node of the *HeinzelCluster* and can run any program locally that does not require a graphical user interface. Communications have been realized with named pipes (but could easily be changed to TCP/IP sockets). *BeeQ* redirects *stdin* and *stdout* to log files and/or reports output directly (much in the fashion of telnet). It maintains a dynamic list of all processes it controls and allows to suspend and kill them. It has a built-in ftp client; processes information which jobs should run concurrently and which not and in what sequence; it can communicate with other instances of *BeeQ* on the cluster for load balancing. We intend to use *BeeQ* for monitoring a "heart beat" function and if necessary restart essential services.

In addition to the obvious need for a reconstruction cluster with HRRT data sets, we intend to use the *HeinzelCluster* also for reconstruction of especially demanding data sets from our ECAT HR and ECAT EXACT scanners (e.g. multi-frame, multi-bed studies). Data transfer from the Sun workstations running CTI's ECAT 7 software has been realized via ftp and gives a performace of 9.5 MB/s. For routine handling, it is necessary to incorporate the cluster reconstruction option into the console's user interface. This can be realized with a CAPP module that sends commands to the cluster via rsh to a redirector module that forwards *stdin* to *BeeQ*.

We currently can communicate with instances of *BeeQ* through a terminal-like tool that runs on the Windows platform from anywhere in the network (utilizing the network protocols TCP/IP or NetBUI), a subset of its functionality will be available as an IDL module (call external).

Reconstructions of HRRT data usually produce image volumes exceeding 50 MB: there is a new challenge to visualize these image volumes and the even more demanding sinograms

(> 300 MB, short integer format) efficiently. With *Vinci*, we have developed an application for Windows NT/2000 (C++ with MFC) that is entirely true color based and takes advantage of some features every modern PC graphic board offers. We have incorporated the reslicing engine that drives the MPITool, [16], which has been the institute's workhorse for visualization so far. *Vinci* also supports the ECAT7 data format and can be integrated into the Syngo frontend on the HRRT console; it only requires moderate hardware resources (memory requirements depend on how many image volumes you want to look at simultaneously) and was designed to run well on midrange laptop systems. In addition, we use parts of its true color engine for online visualization of reconstruction results.

III. Results

With *BeeHive* over Myrinet and HRRT data we are quite close to the theoretical maximum speedup of the cluster while having the benefits of a simple scheduling approach. However, there is still some work ahead to speedup the preprocessing steps: we have started work on running FORE on more than one CPU. This is even more important for the more computationally demanding FORE-J, [17], and FORE-X, [18], flavours which we hope to evaluate for HRRT data in a collaboration with M. Defrise and X. Liu of the Department of Nuclear Medicine, Free University Brussels, Belgium.

With approach (a) of our strategy to accelerate OSEM3D one cluster node already performs in the range of an Alpha EV67. It remains to be seen how much more acceleration approaches (b) and (c) will provide.

IV. Conclusion

HeinzelCluster with *BeeHive* over Myrinet is a suitable platform for several reconstruction strategies, especially for HRRT data. Future work with this platform might include list-mode reconstruction with correction of motion artefacts.

V. Acknowledgments

S.V. thanks M. Schmand, M. Lenox, J. Bao, P. Luk, M. Casey and R. Nutt for fruitful discussions and a most stimulating environment during S.V.'s stay at CTI.

VI. References

[1] K. Wienhard., M. Schmand, M.E. Casey, K. Baker, J. Bao, L. Eriksson, W.F. Jones, C. Knoess, M. Lenox, M. Lercher, P. Luk, C. Michel, J.H. Reed, N. Richerzhagen, J. Treffert, S. Vollmar, J. Young, W.D. Heiss, R. Nutt, "The ECAT HRRT: Performance and First Clinical Application of the New High Resolution Research Tomograph.," IEEE MIC, Conference Record, 2000.

[2] P. Schmidlin, M.E. Bellemann, G. Brix, "Iterative reconstruction of PET images using a high-overrelaxation single-projection algorithm," *Phys. Med. Biol.* 42: 569-582, 1997.

[3] S. Vollmar, W. Eschner, K. Wienhard, U. Pietrzyk, "Iterative reconstruction of emission tomography data with a-priori-information," *Conference proceedings of the 1998 IEEE nuclear science symposium and medical imaging conference*, Toronto 1998.

[4] M. Defrise, P.E. Kinahan, D.W. Townsend, M. Sibomana, D.F. Newport, "Exact and Approximate Rebinning Algorithms for 3-D PET Data," IEEE Transactions on Medical Imaging, Vol. 16, No. 2: 145-158, 1997.

[5] S. Vollmar, K. Wienhard, "BeeHive: Cluster-Computing im NT-Netzwerk", *Conference Proceedings of the 16th DV meeting of the Max-Planck-Society*, Göttingen, November 1999.

[6] S. Vollmar, M. Lercher, C. Knöss, C. Michel, K. Wienhard, W.D. Heiss, "BeeHive: Cluster Reconstruction of 3-D PET Data in a Windows NT network using FORE", IEEE 2000, Lyon.

[7] Intel Corporation, Intel Architecture Optimization Reference Manual, Order Number: 730795-001.

[8] Myrinet networking hardware, http://www.myri.com.

[9] C. Michel, M. Schmand, X. Liu, M. Sibomana, S. Vollmar, C. Knöss, M. Lercher, C. Watson, D. Newport, M. Casey, M. Defrise, K. Wienhard, W.D. Heiss, "Reconstruction strategies for the HRRT", IEEE 2000, Lyon.

[10] D. Schmidt, "The ADAPTIVE Communication Environment (ACE)", http://www.cs.wustl.edu/~schmidt/ACE.html.

[11] c't, 16/99, p. 152, http://www.heise.de/ct

[12] Myricom, Inc., "The GM Message Passing System", http://www.myri.com/scs/GM/doc/gm.pdf

[13] K. Dowd, C. Severance, "High Performance Computing", O'Reilly, 2nd edition 1998.

[14] LAM/MPI Org, University of Notre-Dame, "Top 10 reasons to prefer MPI over PVM", http://www.lam-mpi.org/mpi/mpi_top10.php.

[15] SGI 1450 overview, http://www.sgi.com/servers/1450/

[16] U. Pietrzyk, A. Thiel, K. Herholz, W.D. Heiss, "A hybrid image registration method employing interactive and automated techniques", Neuroimage, 7:789, 1998.

[17] M. Defrise, X. Liu, "A fast rebinning algorithm for 3D PET using John's equation", Inverse Problems 15 1047-1065, 1999.

[18] X. Liu et al, "Exact Rebinning Methods for 3D PET", IEEE Trans Med Imag MI-18 657-664, 1999.